**Project handbook**

**I. Introduction**

This project aims to simulate a real-life e-commerce big data development process in enterprise production environment. The complete process to establish a data warehouse contains the development of data collection pipeline, DWH modeling, KPIs accomplishment and visualization. There are two data sources, buried point data comes from users’ interaction with client application, which tracks user behavior. And business data which records online transaction history are stored in business servers. My objective is to develop a whole process data pipeline and warehousing e-commerce business data, satisfy back end users and provide insights to product manager.

**Objectives:**

1. Establish multiple data sources collection pipeline.
2. Model and develop multilayer user behavior data warehouse.
3. Model and develop multilayer businesses data warehouse.
4. Analyze KPIs from business needs (, , etc.)

**KPIs:**

Users’ buried point behavior KPIs:

Daily/weekly/monthly activity

Daily new users

3, 5, 7, 14 days retene users

Silent user

Lost user

Three weeks consecutively active user

Business KPIs:

Gross Merchandise Volume (GMV)

New added convert

User activity convert

Two months repurchase rate

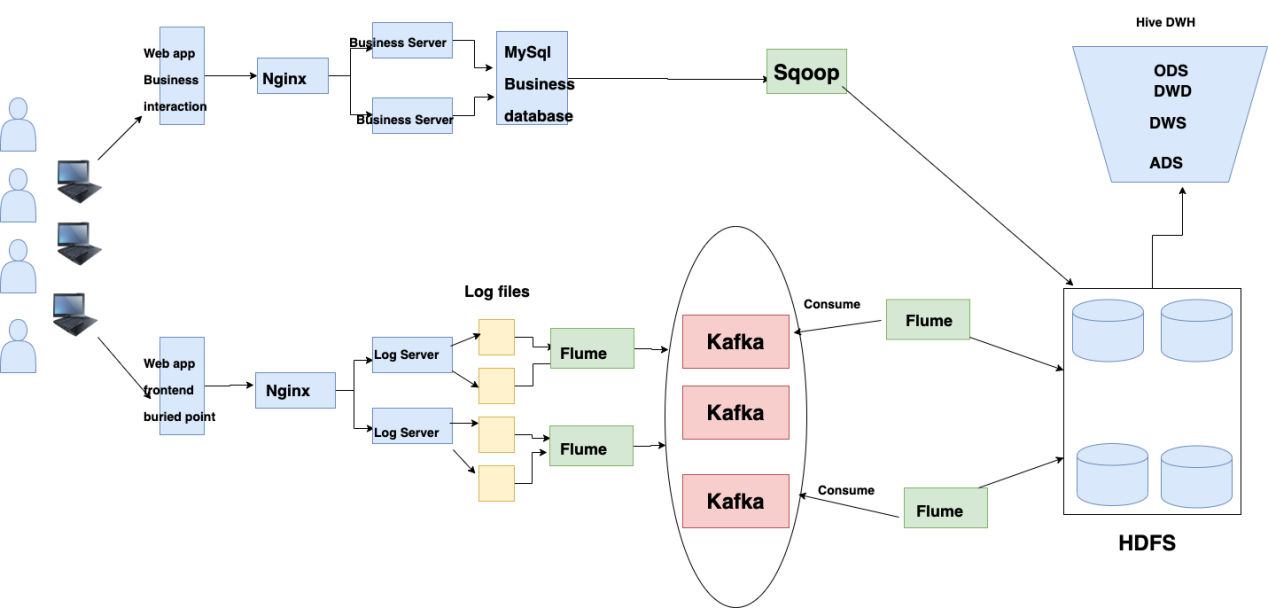
**II. Structure engineering**

**Data sources**

Business data: data such as order, transaction, commodities, payment generated in the business process. Business data usually stored in relational database (MySql, Oracle).

Buried point data: data generated during users’ interaction with application client, include browse history, click record, like, comments, etc.

**Framework**



Data collection pipeline

Buried point data collection: Flume (producer) + Kafka + Flume (consumer) collect log data, cleaning raw data to exclude dirty records and blank records, partition logs into start and event topic and write into Kafka message queue. Kafka is used to balance loads and cut peaks. Flume consumer use file channels to consume Kafka message and write data into HDFS.

Businesses data collection:

Businesses data are structured bulk data recorded in MySql business server, use Sqoop to transfer from business database into HDFS.

**III. Data Explanation**

**Buried point data**

Two types of logs are collected through pipeline: start log and event log. Start log tracks users’ opening records of app, event logs record users’ detail usage history during the browse of the app.

Start log:

|  |  |
| --- | --- |
| **Key** | Value |
| entry | push=1，widget=2，icon=3，notification=4, lockscreen\_widget =5 |
| open\_ad\_type | Open-screen advertising type: native ad=1, interstitial ad=2 |
| action | Load status: 1= success, 2=fail. |
| loading\_time | Loading time |
| detail | Error code (blank if success) |
| extend1 | Error message (blank if success) |
| en | Event type: start |

Event log:

In order to easier the decomposition of event log records, event log is divided into two types of fields, common field consists of the users’ client information, and event fields record users’ detail operation. The following tables explain the content of event logs in detailed.

Event log - common fields

|  |  |
| --- | --- |
| **Key** | **Value** |
| **mid** | **Device id code.** |
| **Uid** | **User id code** |
| **vc** | **Version code.** |
| **vn** | **Version name** |
| **l** | **System language** |
| **sr** | **Message channel.** |
| **os** | **Operating syste** |
| **ar** | **User operation region** |
| **md** | **Device model** |
| **Ba** | **Device brand** |
| **hw** | **Height and width of screen** |
| **t** | **Client time stamp** |
| **nw** | **Network model** |
| **ln** | **longitude** |
| **la** | **latitude** |

Event log - event fields

Event: loading

|  |  |
| --- | --- |
| **Key** | Value |
| action | Start loading=1, load succeed=2, load file=3 |
| loading\_time | **Loading time in ms** |
| loading\_way | 1: read from cache, 2: read from server |
| extend1 | **Extended filed 1** |
| extend2 | Extended field 2 |
| type | **automatic=1, hand loading=2, load from bottom=3.** |
| type1 | Loading error code (blank if succeed) |

Event: display

|  |  |  |
| --- | --- | --- |
| **Key** | | Value |
| action | show product=1, click product=2 | |
| goodsid | Product ID | |
| place | Product order | |
| extend1 | Show type (first exposure=1, second exposure=2) | |
| category |  | |

Event: product detail

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| entry | Entry method: app=1, ad push=2, recommendation=3 | |
| action | Start loading=1, loading success=2, load fail=3, exit=4 | |
| goodsid | Product ID | |
| show\_style | Product show patter: 0= no image, 1= single image, 2= two images,3= three images ... 5= 5 images | |
| news\_staytime | Stay time on the page. | |
| loading\_time | Loading time on product detail page. | |
| type1 | Loading error code (blank if succeed). | |
| category | Product category. | |

Event: ad

|  |  |
| --- | --- |
| **Key** | Value |
| entry | Product page=1, app page= 2, product detail page= 3 |
| action | Request ad=1 cache ad=2  show ad=3 click ad=4 |
| content | Status: succeed=1, fail=2 |
| detail | Error code (blank if succeed) |
| source | Ad source: admob=1 facebook=2  Google =3 VK=4 |
| behavior | Initiative request=1   Passive request=2 |
| News\_type | Type: 1- text & picture 2- picture 3- jokes 4-GIF 5- video  6-questionaire 7-text 8-video & image  9-GIF+text  0-other |
| show\_style | Content style: text=6, one image=1  three images=4 two images=2 images + text=3 |

Event: notification

|  |  |
| --- | --- |
| **Key** | Value |
| action | noti generate=1, noti pop up=2, click noti=3 |
| type | warning=1, weather forecast=2, message=3 |
| ap\_time | App client time |
| content | Extend field |

Event: user front activity

|  |  |
| --- | --- |
| **Key** | **Value** |
| push\_id | Message ID |
| access | 1.push 2.icon 3.other |

Event: user back end activity

|  |  |
| --- | --- |
| **Key** | Value |
| active\_source | 1=upgrade, 2=download, 3=plugin\_upgrade |

Event: comment

|  |  |
| --- | --- |
| Key | Value |
| comment\_id | Individual comment id |
| User\_id | Individual comment id |
| p\_comment\_id | Reply comment id |
| content | Comment content |
| Add\_time | Comment time |
| reply\_count | Reply amount |

Event: like

|  |  |
| --- | --- |
| Key | Value |
| **Id** | Individual like history id |
| user\_id | Individual user ID |
| target\_id | Liked product ID |
| type | 1=like answer, 2=like comment, 3=like product, 4=like reply |
| add\_time | Liked add time. |

Event: favorite

|  |  |
| --- | --- |
| Key | **Value** |
| **Id** | Individual like history id |
| user\_id | Individual user ID |
| target\_id | Liked product ID |
| type | 1=like answer, 2=like comment, 3=like product, 4=like reply |
| add\_time | Liked add time. |

Event log data structure

1540934156385|{

"ap": "gmall",

"cm": { //Common Filed

"uid": "1234",

"vc": "2",

"vn": "1.0",

"la": "EN",

"sr": "",

"os": "7.1.1",

"ar": "CN",

"md": "BBB100-1",

"ba": "blackberry",

"sv": "V2.2.1",

"g": "abc@gmail.com",

"hw": "1620x1080",

"t": "1506047606608",

"nw": "WIFI",

"ln": 0

},

"et": [ //Event Field

{

"ett": "1506047605364", //App client time stamp

"en": "display", // event name

"kv": { //event, showed in key-value format

"goodsid": "236",

"action": "1",

"extend1": "1",

"place": "2",

"category": "75"

}

},{

"ett": "1552352626835",

"en": "active\_background",

"kv": {

"active\_source": "1"

}

}

]

}

}

**Businesses data**

Businesses data are recorded structurally in MySQL database, on business server.

Order info

|  |  |  |
| --- | --- | --- |
| **Key** | | Value |
| id | Order ID | |
| total\_amount | Order amount | |
| order\_status | Order status | |
| user\_id | User ID | |
| payment\_method | Payment method | |
| out\_trade\_no | Payment serial number | |
| create\_time | Payment creation type | |
| operate\_time | Operation time | |

Order details

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | Order number | |
| order\_id | Order ID | |
| user\_id | User ID | |
| sku\_id | Product ID | |
| sku\_name | Product namr | |
| order\_price | Product price | |
| sku\_num | Product amount | |
| create\_time | Creation time | |

Product master

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | Product id | |
| spu\_id | spuid | |
| price | Product price | |
| sku\_name | Product name | |
| sku\_desc | Product description | |
| weight | Weight | |
| tm\_id | Brand ID | |
| category3\_id | Category ID | |
| create\_time | Creation time | |

User master

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | User ID | |
| name | User name | |
| birthday | Date of birthday | |
| gender | Gender | |
| email | Email | |
| user\_level | Membership level | |
| create\_time | Creation time | |

Product category class I

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | id | |
| name | name | |

Product category class II

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | id | |
| name | name | |
| category1\_id | Category I ID | |

Product category class III

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | id | |
| name | name | |
| Category2\_id | Category II ID | |

Payment history

|  |  |  |
| --- | --- | --- |
| Key | | Value |
| id | Payment | |
| out\_trade\_no | Trade number | |
| order\_id | Order number | |
| user\_id | User ID | |
| alipay\_trade\_no | Payment serial number | |
| total\_amount | Payment amount | |
| subject | Transaction content | |
| payment\_type | Payment type | |
| payment\_time | Payment time | |

**IV. DWH modeling and development**

**Architecture**

In this project, DWH is stratified into 4 layers to implement functions and satisfy different business demands. In the real production environment, it is not always necessary to stratify DWH into exactly 4 layers and assign the same names. The real-life architecture design should be adjusted based on business needs.

Operation Data Store (ODS): store original data (data collected directly by pipeline).

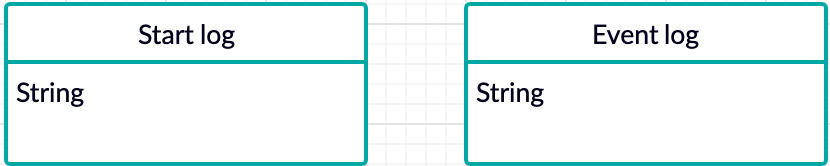
Data Warehouse Detail (DWD): decompose, structure and cleansing original data, transform unstructured data into relational tables or dimensional models (star models in this case).

Data Warehouse Service (DWS): mild summary based on DWD layer, this aims to provide aggregations (wide tables) for KPIs and applications.

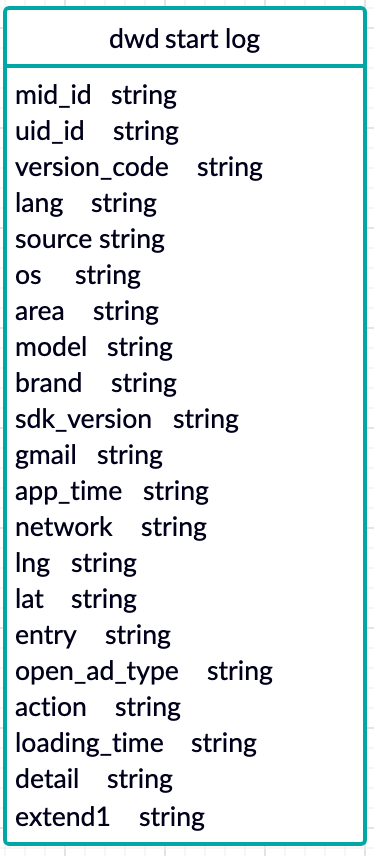
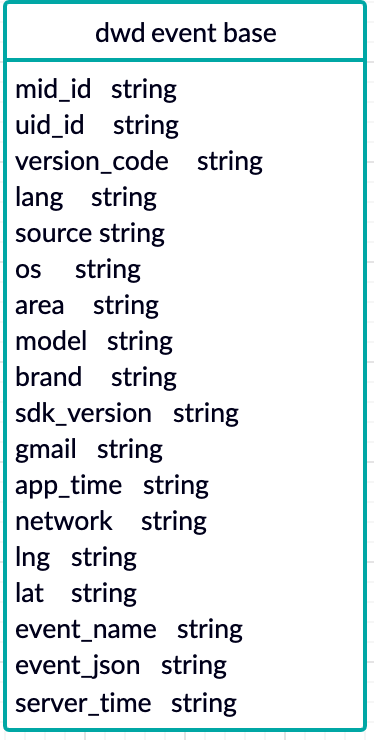
Application Data Store (ADS): provides data for applications and reports.

**Buried point data modeling**

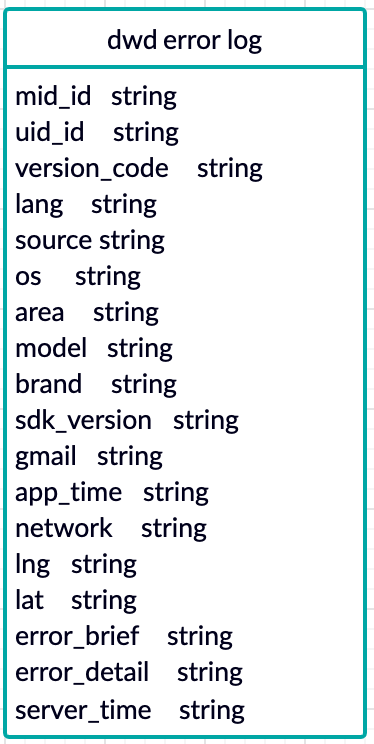
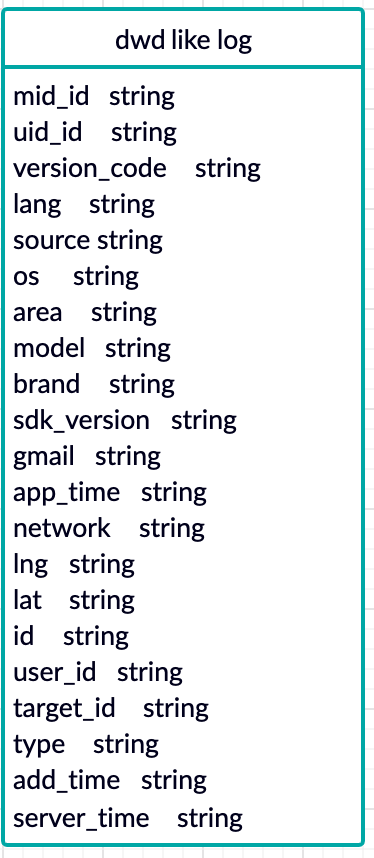
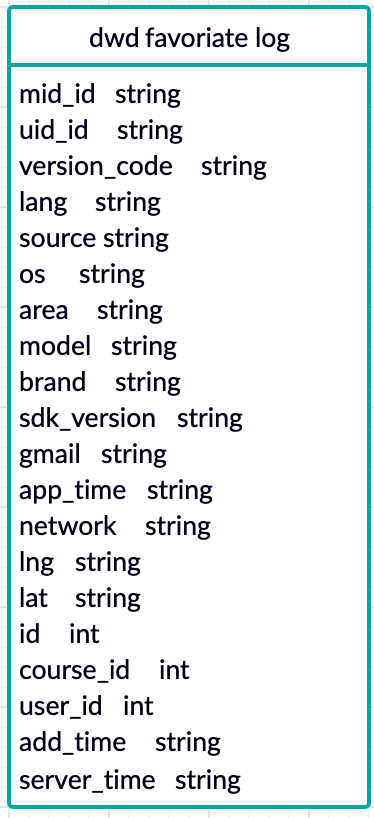
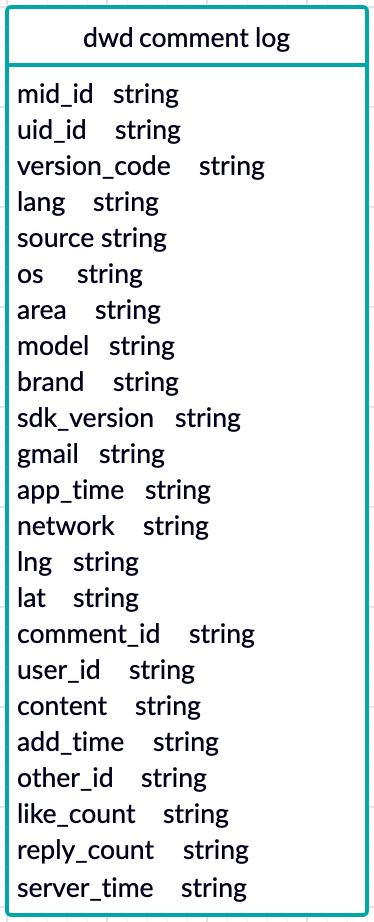
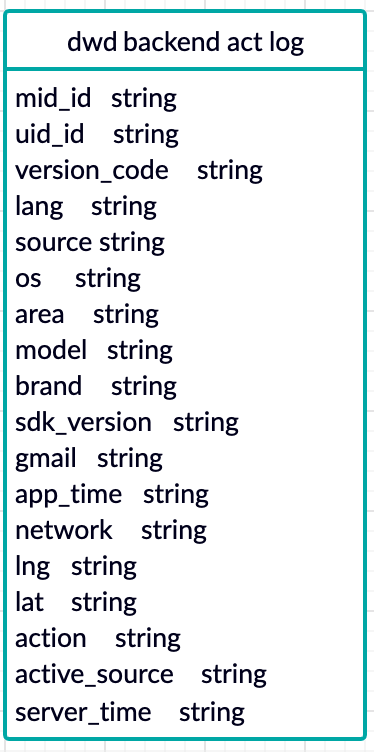
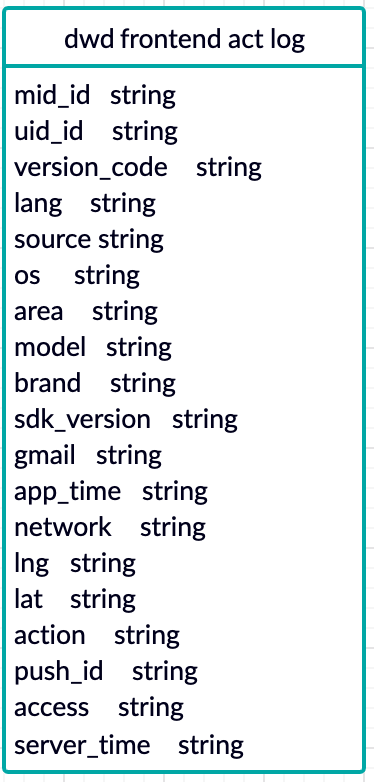
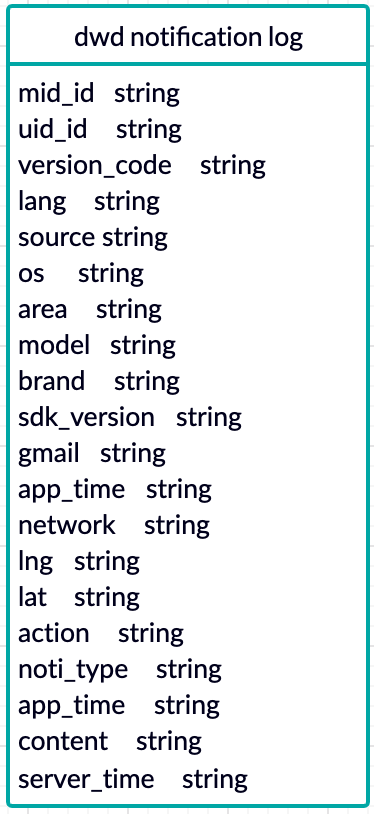
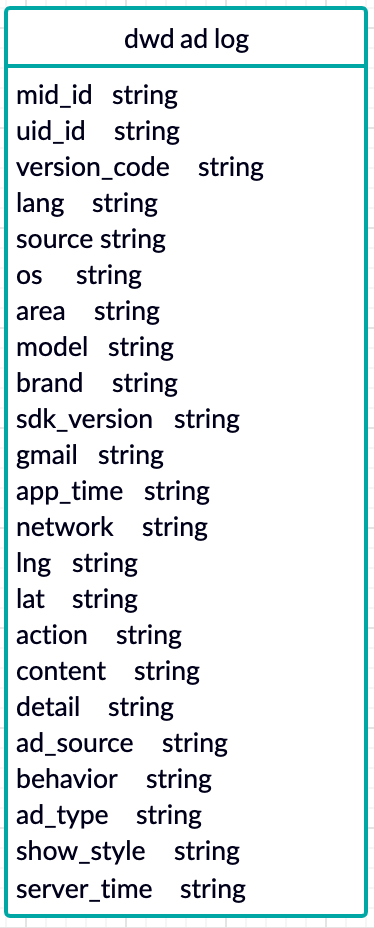
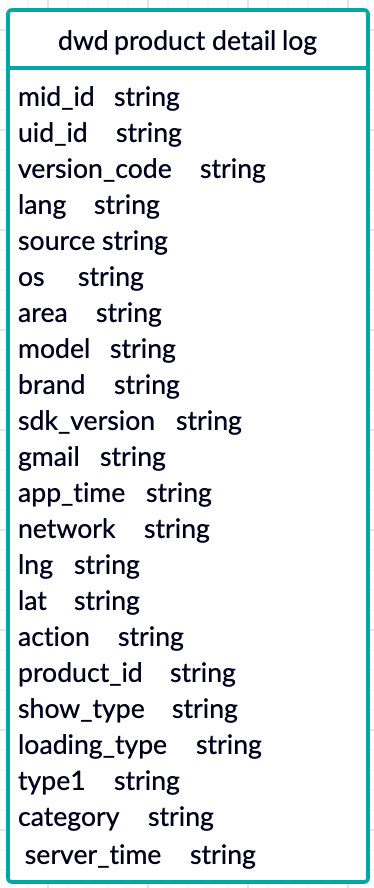
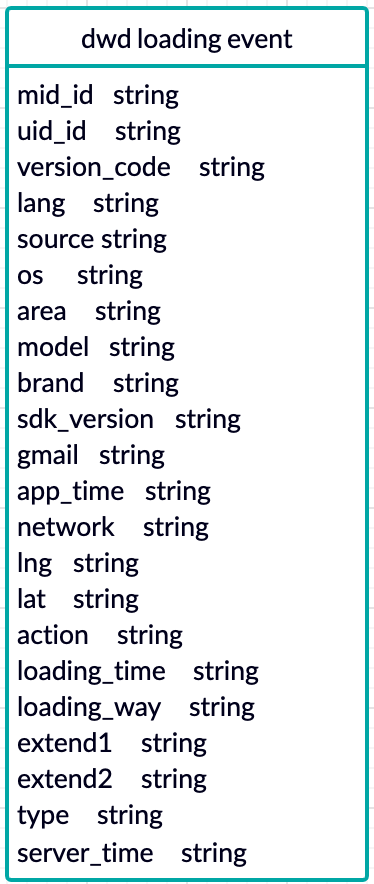
ODS: original partitioned data from Kafka, two long tables with only one column.



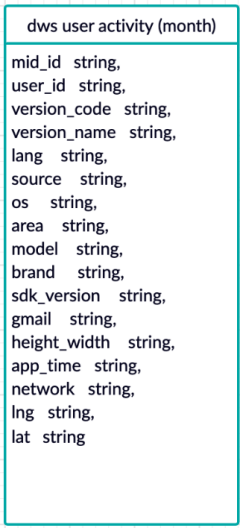
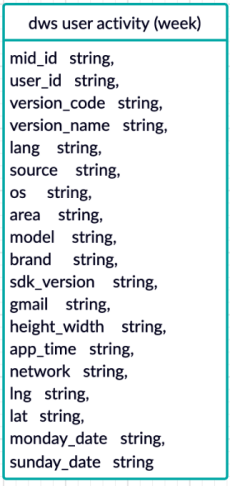
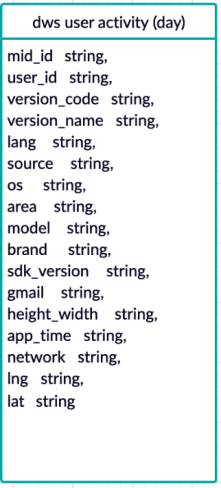
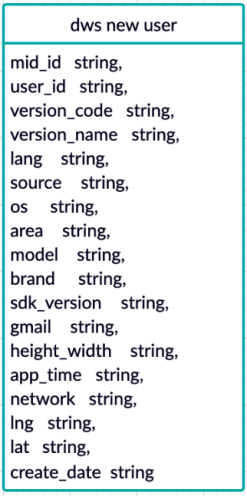
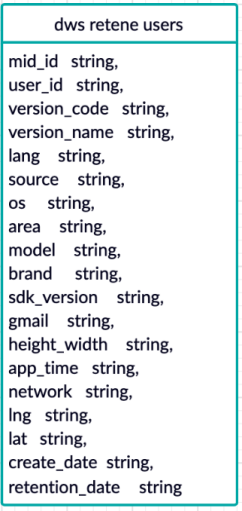
DWD: data cleansing, ETL, remove duplicates, decompose DOS json string into multiple event tables. Event logs are decomposed from event base table.

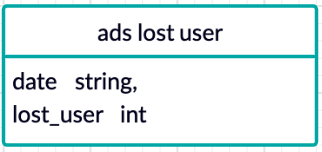
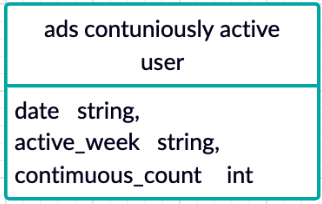
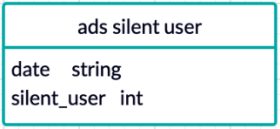
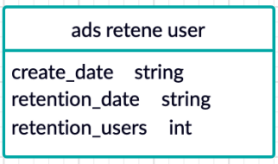
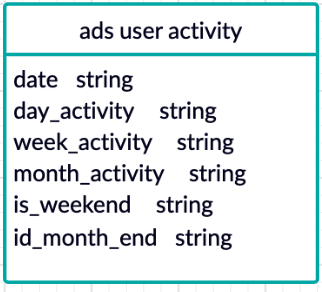
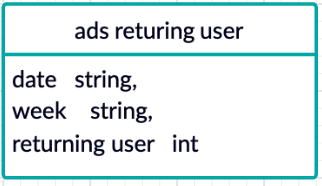
Event log detail tables are decomposed from event base table, there are 9 event topics in total.



DWS: mild aggregation based on business needs, usually contains wide tables.

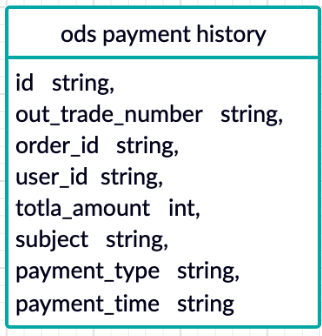
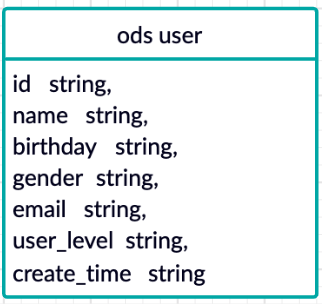
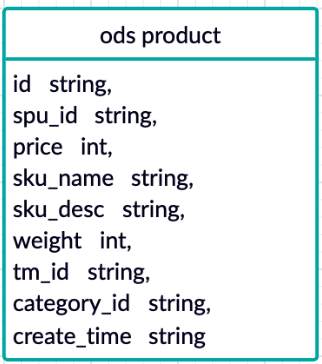
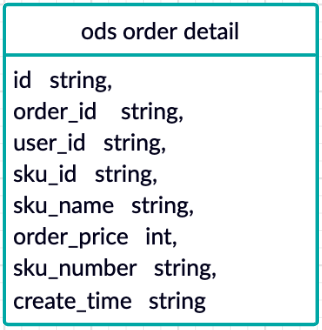
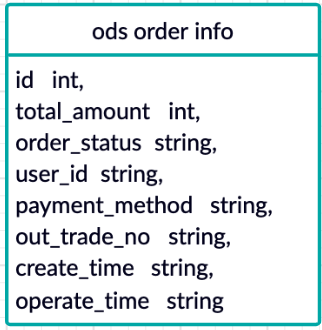
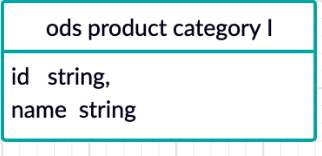
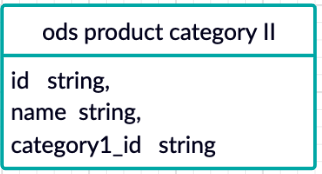
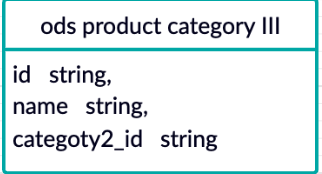
 

ADS: provides result data for data products.

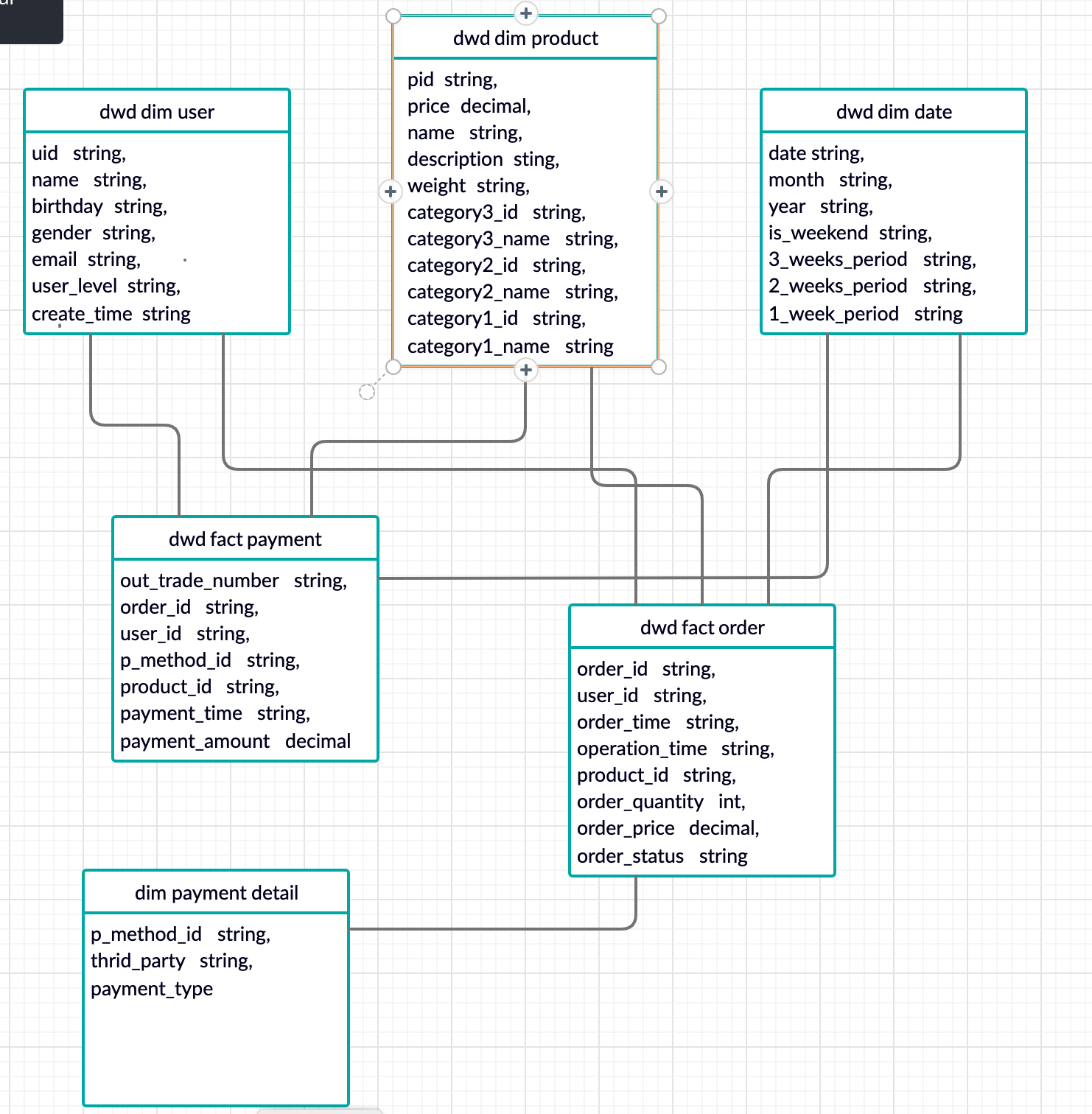
 

**Business data modeling**

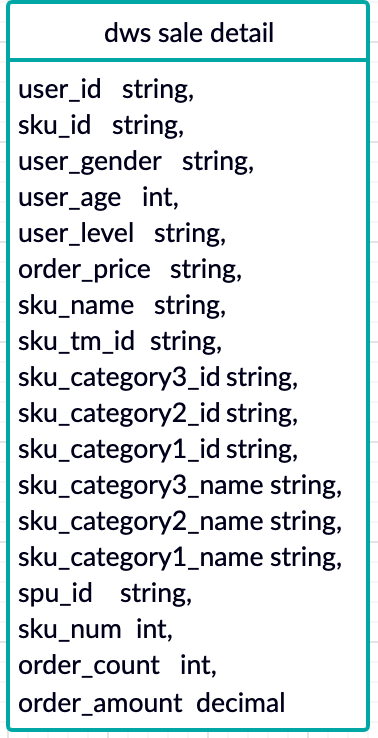
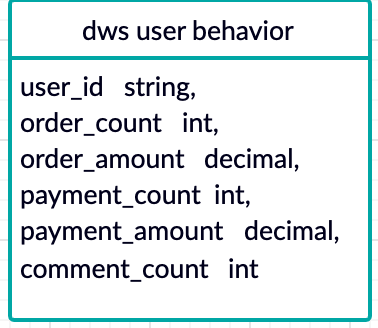
ODS: original business data from business servers

DWD: star models



DSW: user behavior wide table and user purchase wide table



ADS: KPIs

